# 首先构建三层架构

1. **创建好UI显示层，BLL(逻辑层),DAL(数据访问层)**
2. **建立好UI层的界面**
3. **创建好Model(模型)和Common(工具类)，model是在其中创建对应sql数据库中的，common创建连接sql数据库的sqlhelper类和加密数据库的MD5类以及其他工具类**

# 对应需要进行的功能，完善DAL和BLL

1. **首先SqlHelper连接数据库**

**通过输入的密码和用户名，判断是否查询的到。在ui层调用DAL，DAL调用BLL，之后判断是否查询的到。**

**1）、其中在SqlHelper中，有两种连接SQL数据库的方法**

**其一：//连接时，必须对应数据库的名称，服务器以及用户名和密码，是静态私有的**

**private static string sqlstr = "server=.;uid=jgw;password=123456;database=学生学习系统管理DB";**

**其二：//还可以再ui层的中APP.config,创建一个<connectionString><数据库连接地址></connectionString>**

**//再调用system.configuration框架，最后调用其方法**

**private static string sqlStr = ConfigurationManager.ConnectionStrings["sqlcon"].ConnectionString;**

1. **在DAL中编程设定分为三步**

**1）、创建sql语句：**

**对于sql语句中的变量我们需要进行参数化，避免出现敏感字符**

string sql = "select adminname from admins where username=@id and password=@pwd ";

//将查询语句的限制参数化

SqlParameter[] sp = new SqlParameter[]

{

new SqlParameter("id",objAdmin.LogInId),

new SqlParameter("pwd",objAdmin.Logirpwd)

};

//还有一种发方法是

string sql1 = "select adminname from admins where username={0} and password='{1}'";//因为密码是字符串必须使用单引号

sql1 = string.Format(sql1,objAdmin.LogInId,objAdmin.Logirpwd);//也是将其限制的参数化

**2）、调用SqlHelper中的查询方法：**

DataRow dataRow = SqlHelper.GetDataRow(sql,sp);

**3）、返回结果：**

**判断连接是否成功和是否有满足查询的结果**

**判断是否接收到信息，或者为null。**

if (dataRow != null)

{

objAdmin.AdminName = dataRow["adminname"].ToString();

return objAdmin;

}

else

{

return null;

}

1. **在ui层调用BLL（因为BLL调用了DAL）**

**在登入界面：**

**1）、验证输入信息：不能为空，对于user不能有字符，需要以用正则来判断**

//先判断是否输入用户名和密码

if (this.txtLoginId.Text==null)

{

MessageBox.Show("请输入用户名");

}

//利用正则函数判断输入是否是正整数

if (!DataValidate.IsInteger(this.txtLoginId.Text.Trim()))

{

MessageBox.Show("请输入正整数");

}

if (this.txtLoginPwd.Text == null)

{

MessageBox.Show("请输入密码");

}

**2）、封装对象：将输入信息**

**创建对应数据库的类来见将其接受，调用对应的方法，当作参数输入。**

AdmisModel ModeL = new AdmisModel();

ModeL.LogInId=Convert.ToInt32(this.txtLoginId.Text);

ModeL.Logirpwd = this.txtLoginPwd.Text;

**3）、后台交互**

**在main中创建对应数据库的全局变量**

Program.model= AdminBLL.Get(ModeL);

**4）、处理交互结果**

if (Program.model != null)//登入成功，关闭登入窗体，显示主窗体

{

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("请输入正确的用户名和密码");

# 3、窗体的嵌入：

**1、首先要创建嵌入窗体的对象**

**FrmAddStudent frmAddStudent = new FrmAddStudent();**

**2、判断容器中是否已经存在了窗体，需要将其关闭**

//判断容器中是否有别的窗体存在，将其他窗体关闭

foreach (Control item in this.panelForm.Controls)

{

if (item is Form)//判断容器中的是否是窗体，因为窗体嵌入，容器变为父级，可以使用里氏变换

{

Form frm = (Form)item;//将其转换为窗体对象，之后将其关闭

frm.Close();

}

}

**3、嵌入窗体**

**其一：将需要嵌入的窗体取消顶格显示，方面放在容器上方**

**其二：将窗体的边框取消掉**

**其三：选择嵌入窗体的父容器(因为这一个学员对应一个照片所以不用让文件夹对象可以多选图片路径)**

**其四：将嵌入的窗体，选择随着容器大小变化**

**其五：显示窗体**

//嵌入窗体的步骤

//1、将子窗体设置成非顶级控件，这样窗体才可以显示在容器上方

objfrm.TopLevel = false;

//2、去掉窗体的边框

objfrm.FormBorderStyle = FormBorderStyle.None;

//3、指定子窗体显示的容器，指定窗体的父级

objfrm.Parent = this.panelForm;

//4、设置窗体随着容器的大小自动给调整窗体的大小

objfrm.Dock = DockStyle.Fill;

//5、显示窗体

this.lblTitle.Visible = false;

objfrm.Show();

# 程序退出前的确认：

1. DialogResult dialogResult = MessageBox.Show("退出询问","确认退出",MessageBoxButtons.OKCancel,
2. MessageBoxIcon.Question );//messageboxbuttons.okcancel用来增加确认和取消按钮，
3. //messageboxlcon.question用来增加一个问号图片
4. if (dialogResult!=DialogResult.OK)
5. {
6. e.Cancel= true; //告诉窗体关闭这个事件取消

# 5、修改密码：

1. **创建DAL方法，用来修改数据库中管理员表格的方法**
2. **判断旧密码输入正确，新密码是否输入，长度是否满足要求，新旧密码是否相等。**
3. **调用DAL方法，修改密码，将修改的密码返回到全局变量Admin的属性中**

# 6、基于泛型集合动态填充下拉框：

添加学院中，在执行此窗体之前，需要读取出来的时所存在数据库中的班级表，将其存放在下拉框中，也就是在窗体的构造函数中调用DAL查询到，之后将其List<>集合放在comboBox.datasoucre，combobox.data

# 7、在添加学员信息里面，选择照片：

1、首先需要创建打开文件夹的对象

2、确定初始目录

3、确定打开文件夹的名字

4、确定可以选择文件的扩展名

5、显示文件夹窗体，同时给定返回值

6、用返回值，判断在文件夹中是否选择图片，选择图片就将其传给PictureBox.Image的路径，取消了文件夹窗体防止

PictureBox.Image=iamge.FromFile(open.filename);没有得到路径而报错。

//创建打开文件夹对象

OpenFileDialog open=new OpenFileDialog();

open.InitialDirectory = @"F:\wallhaven壁纸下载地";//打开文件夹的初始目录

open.Title = "选择图片";//打开对话框的名字

open.Filter = "所有文件|\*.\*|\*.jpg|";//可以选择文件的扩展名

DialogResult result= open.ShowDialog();//通过打开文件夹的返回值

if (result==DialogResult.OK)//当文件夹取消时，不给图片空间路径

{

//则将选择图片的路径给到图片控件

pbStu.Image = Image.FromFile(open.FileName);

}

# 8、保存学员对象：

1、首先要判断，在添加学生数据的ui中，所有必填的学生信息已填写，生日对应相应的身份证信息，并且电话号码，和卡号必须是正整数，数据库中没有对应新添加的学生的身份证号和卡号。

2、将所有学生信息，封装在student的对象类中，要将图片信息，转化为二进制的字符串信息，然后存入到数据库中去。

# 9、查询学员信息，可以更改删除学员信息

1、按照班级查询学员信息，按照学员姓名和学员学号进行排序。对于学生的排序，采用的是Icommparer接口。

在C#中，IComparer是一个接口，用于比较两个对象的大小。它通常用于对集合进行排序，例如List<>和Array。  
要使用IComparer，您需要实现该接口并提供一个Compare方法，该方法接受两个对象作为参数，并返回一个整数值，指示它们的相对顺序。如果第一个对象小于第二个对象，则返回负整数；如果第一个对象大于第二个对象，则返回正整数；如果两个对象相等，则返回零。

//姓名排序

class NameDESC : IComparer<StudentModel>

{

public int Compare(StudentModel x, StudentModel y)

{

return y.StudentName.CompareTo(x.StudentName);

}

}

//班级排序

class StuIDDESC : IComparer<StudentModel>

{

public int Compare(StudentModel x, StudentModel y)

{

return y.StudentID.CompareTo(x.StudentID);

}

}

if (this.dgvStudentList.RowCount == 0) return;

stulist.Sort(new StuIDDESC());

This.dgvstudentlist.Refrseh();重新排序

1. 对于选定的信息进行修改，利用到DataGridVIew中的

DataGridVIew.CurrentRow.Cell[“studentid”].value.tostring()；来获得选定行数据的某个列单独的数据。

2、修改学员信息，已知学号是主键，无法修改，需要将其所在的控件的属性Enabled=false,不启动该控件；在修改之前还需要判断是否输入了相对应的数据，判断重新输入的学生身份证是否在数据库中，已存在；修改之后，需要再次执行相对应的查询，保证数据的跟新。

3、